11th ICCRTS

INTERNATIONAL COMMAND AND CONTROL
RESEARCH AND TECHNOLOGY SYMPOSIUM
"Coalition Command and Control in the Networked Era"
September 26-28, 2006
Cambridge, UK

Life-Cycle Support for Information Systems Based on Free and Open Source Software

I-136

Richard Carbone and Robert Charpentier
Defence Research and Development Canada

DRDC Valcartier
2459 Pie-XI Boulevard N
Val-Bélair, QC
Canada G3J 1X5

Point of Contact: Robert Charpentier
(T) 418-844-4000 x 4371
(F) 418-844-4538
E-mail Address: Robert.Charpentier@drdc-rddc.gc.ca

Track Topic:
"C2 Concepts and Organizations"
"Lessons Learned"
"C2 Architecture"

Life-Cycle Support for Information Systems Based on Free and Open Source Software

Richard Carbone and Robert Charpentier
Defence Research and Development Canada - Valcartier

Abstract

Free and Open Source Software (FOSS) has been growing constantly in importance and expanding in many software architectures the world over. Increasingly, open source and commercial software share the same ground and will "hybridize" each other in complex architectures. This phenomenon can be observed in both civilian and military information systems. In April 2005, a technical workshop was organized under the auspices of the TTCP-JSA (Joint Systems and Analysis) to determine the role of FOSS in military computing. The key findings of this workshop confirm that FOSS is a very useful technology in military architectures and its importance should constantly increase in the future. However, software support models for FOSS and commercial packages are significantly different and have been a puzzlement to many users and their organizations. In this article, the four basic support strategies that are applicable to software maintenance are described, and some guidelines are proposed for ranking them in a given application context. It is also recommended to adopt them in a progressive strategy starting with support offered by vendors and by competent consultants before adhering to a FOSS consortium. As a last resort, tasking full responsibility of the maintenance of the code can be envisaged.

Acknowledgements

The authors wish to thank Norbert Haché (Directorate of Maritime Maintenance and Support – Canadian Forces) for his guidance in this study. We would also like to thank those who contributed greatly to the technical review of this article by offering many useful comments and suggestions that helped in validating the content of our study: Mr Louis Bastarache (IEEE-Québec), Dr Denis Poussart (Retired Professor – Université Laval), Maj Peter Kvas (IMST- Canadian Forces), Maj J.P. Mellway (IM Policy- Canadian Forces), Mr Terry Bollinger (Mitre). Also, we would like to address special thanks to the heads of delegations for their contribution to the TTCP-JSA workshop: Dr Iain Macleod (Aus), Dr Sue Haines (UK), and Dr Douglas Maughan (USA).

1.0 Introduction

This article is one of a series of publications on Free and Open Source Software (FOSS) that Defence Research and Development Canada is preparing for the Canadian Forces (CF). In our first report presented last year at the 10th ICCRTS, we concluded that FOSS is a very useful technology since it allows the adoption of open standards in military computing; it can lead to some significant cost savings and higher quality information systems [1].

In April 2005, we held an international workshop in Ottawa on the use of FOSS in military computing. The workshop was attended by 25 scientists from four countries (Australia, Canada, the United Kingdom and the United States). Contributions to the workshop included technical presentations, white papers or policy presentations, and scientific articles relevant to FOSS. Chapter 2 gives an unclassified overview of this event.

However, the main concern expressed by the Canadian military community is related to the adoption of an optimal support strategy for FOSS-based systems, which could be significantly different from traditional systems based on proprietary software. So, the focus of Chapter 3 is to describe the various maintenance strategies that could be used in life-cycle support for FOSS-based systems. Our objective is to critically review each of them in order to help project managers understand the impact of a long-term commitment to FOSS in military computing. Then in the next chapter, we will examine the support options for Linux throughout the life cycle of the Canadian frigate information system, which is estimated to last approximately 20 years!

2.0 The Value of FOSS in Military Computing

2.1 Overview of Technical Discussions at the Ottawa Workshop

The workshop agenda is reproduced at Appendix A. The 2½ days of work were busy and productive. The quality of the presentations was very high and the diversity of topics covered was also very good. Some of the material presented by the four participating nations was more strategic in nature, and is summarized in the following section (2.2). Some of the material described actual implementations of FOSS in military systems, allowing the extraction of very informative "lessons learned and best practices," as presented in section 2.3. Also, some pending issues were raised during the discussions, and are listed in the last section of this chapter (2.4).

2.2 Key Findings about FOSS Usage in TTCP Nations

2.2.1 All TTCP nations are experimenting with FOSS

We knew that civilian communities in Australia, Canada, the United Kingdom and the United States were actively testing and integrating FOSS into their information systems; however, the Ottawa workshop confirmed that military communities are also progressively adopting FOSS.

2.2.2 FOSS is a useful technology for military computing

The general perception of FOSS is very favourable, and is supported by strong evidences such as:

- successful R&D projects [A.5, 6,10,11,12] (see Appendix A for list of talks)
- evolution of FOSS towards more secure and reliable systems [A.3, 4,5,16,17]
- evolution of FOSS projects towards supported software products [A.13, 14,17]
- successful deployments in military systems and a few operational theatres. [A.5, 6,11]

2.2.3 Use of FOSS should be increasing in military computing

In similar fashion to the civilian world, the advantages of using FOSS often outweigh the disadvantages. The growth of FOSS is expected to be incremental and should not be fundamentally disruptive for military organizations. FOSS appears to support military requirements for more secure, robust and maintainable systems, with the added benefit of adaptability, code portability, and freedom from vendor lock-in. [A.3, 4, 5, 11, 15, 16, and 18]

2.2.4 R&D communities should demonstrate leadership in FOSS experimentation

The R&D community has an important responsibility to initiate projects that demonstrate the strategic value of FOSS and help make a clearer business case for the armed forces of TTCP countries. Such exploratory activities could lead to a risk reduction in the technological evolution of our nations. The following section includes some lessons learned on military projects using FOSS, which could be further expanded on with more experimentation and systematic information sharing between our allied nations. [A.11, 12, 13]

2.2.5 Open Source can prevent loss of research results

Some R&D currently done by governments can be transferred to industry or end-users (i.e., clients of our research programs); however, a very significant portion of our work is simply lost because no one can in fact economically recover the research outcomes. Through technology giveaway, the Open Source paradigm can play a role in preventing this waste. The last century of scientific publication contains numerous examples of research results that were successfully recovered to achieve new goals and products that were inconceivable to the original research team. Modern R&D organizations around the world are considering this new publication opportunity (i.e., software sharing) to complement and support the publication of traditional documents. Research agencies in TTCP countries can very likely include this delivery mechanism in their programs. It should be noted that FOSS implementations are often highly valued by TTCP armed forces – frequently more highly valued than scientific articles that are in many instances considered too academic and not readily exploitable. [A.13, 18]

2.3 Lessons Learned and Best Practices

2.3.1 Adoption of open standards offers fundamental advantages and should be prioritized

Software development using open standards is leading to improved interoperability with other systems (either national or international), easier component substitution (very useful when a better module becomes available), and better security through the use of standard security solutions. When standards are lacking, it may be preferable to invest in the development of new open standards rather than integrating custom (or proprietary) solutions. [A.6, 11]

2.3.2 Adoption of FOSS must be based on its intrinsic (well understood) advantages

Unbiased assessment of FOSS and COTS components is mandatory to leverage on best-of-breed software. Both technical and legal perspectives must be examined. Complementary investments are typically required for FOSS as well as for COTS in order to evaluate, adapt, improve, certify, etc. [A.6, 11]

2.3.3 Hybrid architectures using GOTS, COTS and FOSS are often the best approach

FOSS is great, although not a complete solution. As mentioned in A.4, “synergistic use of open and closed source should be our goal in order to achieve the best overall architecture.”[A.4, 6]

2.3.4 Access to source code has proven very beneficial and practical

With access to source code, it is easier (i) to understand how the applications work, (ii) to add capabilities and (iii) to fix problems when they are detected. Verification and validation (V&V) techniques have greatly improved over the last 12-18 months and many good tools are currently available to both check and harden software prior to its integration. Understanding (and recovery) of software architectures will remain difficult, especially when a complex system of systems is under study. [A.4, 6, 7, 8, 15]

2.3.5 Configuration Control is required to comply with licensing constraints

The development of software systems and specific applications raises the requirement for appropriate engineering processes and practices to be followed during the course of development. The most important of these is to have appropriate software revision control throughout the development process if some original code is produced and is not intended to be shared (i.e., original IP). The revision control data must remain available after the development cycle is finished in order to prove where the code came from. Without a complete history of the code development captured in a software revision control system, there is a high risk of the new code falling under other licence models such as the GPL (or other similar licence model). [A.11, 19]

2.3.6 Software certification is useful but long, expensive and ephemeral for exposed systems

Similar to what happened with COTS software, some FOSS projects are now engaging in a rigorous certification process (such as Common Criteria approaches), which can certainly improve the level of confidence in a given product and would open up some advanced application domains. Criticisms were expressed at the TTCP Ottawa workshop about the time and cost (and associated complexity) of such certification. Furthermore, certification may be very ephemeral when the deployed software is exposed to the Internet or other hostile agents on the network. In some instances, security measures may be technically acceptable but become intolerable to the end-user during tests (e.g. slowdown). On the other hand, there is no real practical alternative to the Common Criteria, and the need to certify software will continue to increase with the proliferation of FOSS in critical infrastructures. [A.4, 9, 17]

2.4 Pending Issues Deserving More Attention

2.4.1 The strategic value of FOSS in military computing is still inadequately understood

The intrinsic limitations of closed source software may be too constraining for many military systems in the future. Even though the closed source strategy appears to be appropriate for the mass market (e.g., domestic or personal users with little or no programming skills), for military systems and government computing in general, access to the source code and the adoption of open standards are obvious advantages. The need for higher reliability and security, greater flexibility and scalability, increased competition in software supplies and direct cost savings will tend to justify considering FOSS throughout the next decade. It is felt by the technical community that the strategic value of FOSS in military computing is inadequately understood. [A.5, 12, 14, 18]

2.4.2 Adjustments may be needed in military organizations to migrate smoothly to FOSS

FOSS adoption involves significant changes in our organizations. First, the acquisition process is very different for FOSS [A.18]. Building and maintaining expertise in software comprehension and improvement will be crucial (see sections 2.3.4 and 2.3.6). When original code is to be developed, adopting a rigorous version control system will be imperative (see section 2.3.5). Training staff to interpret licences will also be mandatory (see section 2.4.4). On the other hand, FOSS offers outstanding opportunities to distribute technology and research results via software sharing that could support the publication of traditional documents (see section 2.2.5).

2.4.3 Cost modelling is clearly inadequate and may well be impracticable

Cost modelling is very difficult for software systems, especially when the period under study is rather long. It is clear that budgets need to be planned differently when FOSS is to be used (less spending on acquisition and more on developing confidence) as opposed to COTS software (more on acquisition and potentially less on maintenance). The increased requirement for highly secure systems will also further disrupt our limited ability to predict costs in the next decade because of our limited experience with certification. It is generally recognized by the scientific community that cost modelling is clearly inadequate and may well be impracticable for some time. Lessons learned from R&D projects can potentially help to better understand this issue. [A.3, 17]

2.4.4 Licence interpretation is complex and requires immediate attention

Many licence models are currently used and may significantly impact the ownership of some original code. The reality of the various licences is complex, and a complementary study is desired by the scientific community to provide legal advice on licensing, property rights, technology transfers and other legal issues. [A.11, 18, 19]

2.4.5 Understanding potentially offensive FOSS

The FOSS development model has proven to be very efficient. This is true for developers with good intentions as well as for those with malicious objectives such as the development of FOSS with offensive capabilities. Furthermore, self-assembling groups of attackers can learn more quickly, explore new attack methods, operate effectively on very small budgets, and co-opt naïve regions of the Internet for more power and automated attack modes. [A.3, 4, 15]

3.0 Life-cycle support of FOSS-based systems

Following the TTCP JSA workshop, the Directorate of Maritime Maintenance & Support requested a complementary study on life-cycle support of FOSS. This was needed to better understand the "long-term" impact of adopting FOSS in the Canadian Navy. So, in this chapter, we will describe the four basic support strategies that are applicable to software maintenance of FOSS and will propose some guidelines on ranking them in a given application context. The first section (3.1) offers some general principles that are useful in planning the support strategy.

3.1 General strategy in FOSS adoption

A- Adoption of FOSS should be progressive

As mentioned in previous studies, "adoption of FOSS can have fundamental and far-reaching consequences on engineering practices, especially if the objective is to contribute actively to an open source project. It is recommended that experience be gained with FOSS as a passive user first, then to become progressively more involved by reporting bugs, suggesting new features, and modifying existing code before engaging in active development within a collaborative project." [2]. In a similar way, the question of support would be better addressed if a progressive strategy were adopted. Fortunately, some life-cycle support models for FOSS are very similar to those used with proprietary software. It is therefore recommended that these familiar models be utilized at the beginning to gain expertise and ensure continuity of service.

B- FOSS offers more options than COTS software in terms of long-term maintenance

Four basic support options are possible with FOSS. They are introduced below in increasing order of complexity:

i) Support offered by vendor or commercial integrator (similar to COTS software)

ii) Support contracted to credible consultants
iii) Support obtained through a consortium of users and developers
iv) Support assumed in-house by the owner of the system.

With commercial packages, life-cycle support is typically based on option (i) with some help from the integrator, and in the case of very large information systems, option (ii). However, options (iii) and (iv) are to a large extent specific to software whose source code is available for maintenance (i.e., configuration management, patches, fixes, etc.) and improvement (i.e., customization of functionalities, new features, code hardening, etc.). This is why FOSS offers more possibilities with regard to maintenance, especially for the long term (beyond commercial viability of a software product).

C- Complementarity of support options should be exploited

With FOSS, support options are not exclusive and they can be mixed and organized to better complement each other. For example, in the case of a problem with the prime option selected, another one can be used as a fallback option. In the case of classified needs not covered by the commercial world, it may be appropriate to hire a cleared consultant or to pursue development in-house. In some other instances where common interests are identified, it may be best to form a consortium to address common needs more efficiently. In the case of very long-term support beyond commercial profitability, it is always possible to do it locally while leveraging some contributions from the FOSS community. The complementarity of the support options is one of the most valuable advantages of FOSS over COTS software.

D- Choices should be assessed and validated on a case-by-case basis

In the next section of this chapter, a more detailed description of each support option is presented and should facilitate selection of the best support option in the context specific to each project. The following section (section 3.3) offers a basic approach to ranking various support options in order to maximize the benefits of a trade-off analysis, done on a case-by-case basis. It is also recommended to consult the lessons learned from previous projects as well as relevant reports from forecasting firms to help validate the ranking. Exchanges of information via international forums (such as TTCP, NATO, or bilateral) can also be of great help. In many instances, pilot projects may be required to confirm the practicality of the solution(s). Licences and other legal issues should be part of the evaluation, since they may preclude application in some life-cycle support options.

3.2 Life-cycle support options for FOSS-based systems

In this section, a more detailed description of the four support options for FOSS-based systems is presented. Even though we have critically reviewed the following descriptions, some of our statements remain essentially qualitative and could only be made more absolute in reference to a specific application. For example, what is considered to be an advantage in general may be inappropriate in a particular project, and vice versa. Furthermore, typical risks, costs and their mitigation can vary greatly from one application context to another.

3.2.1 Vendor support should be considered first (as done for COTS software)

What it is: Many vendors integrate, debug, and sell FOSS packages that are very similar to various COTS software suites. The main differences are that the FOSS suites are not proprietary, the source code is available, and the cost is typical lower than their commercial equivalents. FOSS resellers normally offer technical support to quickly and accurately resolve end-user issues via a knowledgebase (for known problems) and multiple forms of technical support.

Pro: This support model is very similar to COTS software
Low usage of internal resources

Efficient leveraging of user experiences and debugging information
Technical training often available
Simpler accountability
May simplify management of component licences via the overriding effects of the integrator's legal terms

Con: Little influence on product evolution
Largely dependent on supplier, who may try to lock-in customers as in COTS software
Support may not be available for less popular FOSS
Variability and/or uncertainty of vendor licences (e.g., distribution limitations)

Typical risks:
Supplier abandoning product (for new ones)
Supplier abandoning business
Supplier forcing specific methodologies/solutions that are incompatible with enterprise policies

Risk mitigation:
Negotiate well-defined SLA (Service Level Agreement)
Be capable of changing product or supplier (as in the case of COTS software)
Use one of the options made possible through the access to source code:
Hire a consultant,
Request support from the developer community, and/or
Use internal resources to support the FOSS-based system

Typical cost drivers: In general, vendors offer support on a "Licence per seat or per server" basis, which leads typically to a global cost that is proportional to the size of the organization. However, the support cost and its basic offering vary greatly from one vendor to another. Novell, for example, allows customers to buy as much technical support as they need (e.g. per server basis, small business support, and premium). On the other hand, Red Hat bundles the technical support agreement into the licence and offers few support packages – basic, standard, and premium. Further discussions of costs and management strategies are available in [3, 4, and 5]. It is more likely that the vendor-support option is one of the two least expensive options, followed by joining a consortium.

Author's assessment: Ideal to reduce anxiety associated with FOSS adoption and to alleviate legal issues by passing them on to a vendor. Quite inexpensive too! Some vendors have established a strong reputation for quality and reliability that brings them to levels comparable to the best COTS vendors.

3.2.2 *Alternatively, consultant support can be sought*

What it is: Many consultant firms offer integration and support services of FOSS components. Their services can range from component selection, integration, testing prior to deployment, configuring and monitoring behaviour, diagnostics, resolving problems, development of specialized features, and ultimately offering long-term maintenance for deployed systems.

Pro: Moderate usage of internal resources
Support strategy can be optimized for specialized requirements (e.g., classified systems)
Technical training can be optimized for local requirements
Access to highly skilled individuals

Con: Could become expensive – must check if economically viable

Additional administrative constraints and overhead to cope with
Dependence on the availability (and maintenance) of competent resources
Must include management of legal issues associated with selected components

Typical risks:

Supplier abandoning business or such services
Obtaining long-term commitment of the resources
Lack of financial resources to sustain the support model

Risk mitigation:

Implement methodologies to integrate the work of multiple contractors
Hire a new consultant firm to replace a supplier abandoning the business
Use one of the other options made possible through access to the source code:
Request support from the developer community and/or
Use internal resources to support the FOSS-based system

Typical cost drivers: In general, consultants offer highly customized solutions that are optimally suited to their clients. The global costs are essentially proportional to the size of the support team being assigned to the contract. The intrinsic inefficiency of this model stems from the fact that little leveraging is made possible by such support outsourcing. Outsourcing to a foreign country that has a lower cost structure (e.g., India) can also be considered in some instances. Globally, the consultant-support option can only be among the most expensive strategies (with self-support – Option 4) since leveraging is not fully exploited.

Author's assessment: Can greatly reduce anxiety associated with FOSS adoption. Small businesses offering FOSS support are proliferating but this approach could prove to be the most expensive strategy to support software.

3.2.3 When experience is gained, joining a consortium becomes appropriate

What it is: Open source communities have a reputation for offering very good support to end-users by answering questions rapidly, correcting bugs, creating tutorials, and regularly improving their packages with new features. This type of support is obtained at no cost but it can be quite variable from one project to another depending on the volunteers available. Typically, large mature projects (e.g., PHP, MySQL, Apache etc.) provide the best services but may offer to sell them in order to generate revenue to keep the project expanding. In such a case, it is an acquisition of services similar to option 1 described above. Some end-user communities also form consortia to exchange developments (and/or support) specific to the business area. Sometimes called peer-to-peer (P2P) support, this includes mailing lists, knowledgebase repositories, Web sites, and chat rooms. A classic example is in the educational world, where many schools share domain-specific software. Similar examples can be found in the health care sector and municipal administrations. In the military world, it is quite realistic to see consortia being developed with allies, TTCP countries, NATO, and nationally with other government departments (OGD) with which the CF are closely connected.

Pro: Much to be gained when well done
Inexpensive or free without vendor lock-in
Could allow collaborative work with allies within a bilateral context such as TTCP or NATO

Con: Some competent resources must be committed and sustained internally

Contributions to the community may be expected, especially when participating in domain-specific consortia (e.g., bug fixes and enhancements released to the consortium)
Improved source code may often have to be made publicly available
Time required understanding the dynamics and in establishing credibility in the community
Documentation may be less user-friendly (somewhat too technical)
Information overload is a danger
Free support services typically entail no legal liability

Typical risks:

Possibility of losing control of consortium or diverging from original objectives
Non-friendly nations could leverage on the consortium work

Associated mitigation: When consortium support becomes inadequate, support can sometimes be bought or subcontracted, as in options 1 and 2, or simply taken on as an internal responsibility, as described in 3.2.4.

Typical cost drivers: Global costs are essentially proportional to the size of the support team. When the user community is broad enough (and well centralized), it may be possible to share support costs to a level that would be sustainable. The main advantage of the consortium-based strategy stems from leveraging internal human resources with the tremendous potential resources of volunteers and other external contributors involved in the consortium. Globally, the option of support-through-consortium is considered quite economical when compared with options 2 (consultant-based) and 4 (self-support).

Author's assessment: Several computer companies financially support the development of FOSS and try to leverage the generous offers from thousands of volunteers. Similarly, governments can certainly gain a lot from consortia but it may require significant changes in current engineering practices given the business model underlying support of software by consortia. This may require government agencies to evolve their practices in order to understand this new way of doing business, including understanding the hierarchy of FOSS projects, finding ways to provide incentives, and delegating some control to volunteers. A list of dominant FOSS consortia (21) is provided at Appendix B.

3.2.4 In some specific cases, it may be best to assume full responsibility for the software

What it is: Some FOSS applications can be integrated and merged with the original software to produce a customized system. Such FOSS-based systems are sometimes maintained solely by the use of dedicated internal resources. This support model has been used for many years by financial institutions such as major North American banks. Similarly, in the military world, it could be appropriate to very tightly control the core Operating System (OS) of a hardened C2IS deployed in a military theatre in order to protect it from an opponent's attack or other subversive exploits.

Pro:

Full control of the technology being implemented and supported
Money is spent locally in building in-house expertise
Implementation specifics contribute to code diversity, which reduces proliferation of automated attacks and allows hardening of critical core components
Can allow evolution towards local needs and specific requirements that are not economically viable
Could be the only viable option for very long-term support (20 years and more)
Packages can be reduced to core functionalities that minimize software bugs, thus reducing defects and potential exploits in deployed C2IS

Con: Core competent resources must be committed and sustained internally
Time required to develop, test and debug custom software components
Often comes down to a few technical developers becoming indispensable
All legal liability for support is assumed internally
All financial charges to be assumed with no leveraging from other users

Typical risks:
Loss of internal resources can be detrimental to support service
Full legal liability to be assumed locally

Associated mitigation: When internal resources are insufficient, support can sometimes be bought or subcontracted as in option 2. A series of practical experiences and much basic advice are offered in the following set of articles: [6, 7, 8, and 9]. Some weaknesses are also listed in [10], providing a useful complement to the aforementioned articles.

Typical cost drivers: Main costs are the human resources needed internally to take full responsibility for the technical and legal aspects. As mentioned for option 2 (consultant-based support), the lack of efficient leveraging makes stand-alone approaches rather expensive and risky. This approach may be justifiable in many instances where security is an issue, or where very specialized systems are needed to address unique requirements such as weapon system controllers.

Author's assessment: Probably the most demanding option, this is the one that gives the best performance for highly specific requirements or for security-critical systems where purchasing or sub-contracting is not readily feasible. However, it could become expensive and legally complex. For practical experiences in the military community, the following articles may be enlightening:

i) Open Source Opens Opportunities For Army's Simulation Systems [11]
ii) Open Source Software: Opportunities and Challenges [12]

3.3 Basic Approach to Ranking Support Options

To assist project leaders and system managers in identifying the most appropriate life-cycle support strategy, in this section we offer a basic approach to ranking the various options. For each phase of the evaluation, a list of relevant issues is proposed and could serve as a basic checklist in the selection process.

Phase 1- Maintenance context must be well defined

This should include an accurate evaluation of:

A - The requirements
e.g. The continuity of services demanded by the organization
The security/reliability requirements
The expected operational life

B - The expertise
e.g. Available in-house
To be developed internally
Available through partners/collaborators/sub-contractors

C - The technical context
e.g. Operating system in use in the organization
CPU in use in the case of OS selection

Special hardware or software requirements
Data and message standards to comply with
Quality of the documentation on the legacy architecture
Most likely technical roadmap for system evolution

D - The legal framework

e.g. Existing policies or constraints within the host organization
Planned redistribution to partners, collaborators, contractors, etc.
Releasability of patches and fixes to the general public

Phase 2- Selection criteria that have a direct impact on the support plan must be prioritized

This should include but not be limited to:

Quality of the documentation of candidate FOSS components/applications
Existing network of expertise readily available in the FOSS packages
Ease of installation/ease of uninstalling the software modules
Warranty
Estimated long-term product viability

Phase 3- All appropriate support options must be considered

Referring to the description of each life-cycle support option given in section 3.2, review each option with reference to your maintenance context as summarized in Phase 1. Clarify risks and associated mitigation strategies applicable to the specific context under study. It may be appropriate to consult lessons learned from previous projects, and to read relevant reports from forecasting firms such as Gartner, IDC and others. Exchanges of information via international forums can also be a great help.

Phase 4- Costs should be estimated

Even though cost estimation in information technology (IT) is intrinsically difficult, some effective methodologies are available to help quantify the total cost of ownership (TCO) and the strategic value of IT infrastructures.

Phase 5- The selected FOSS-based architecture should be prototyped

Running a pilot project is often mandatory to confirm functionalities, measure performance, and confirm user acceptance. It can also be used to assess the performance of the supplier or local support services, assess supplier commitment to support, and ensure evolution of the product. Prototyping activities could also greatly help a less experienced team to learn about the maintenance *modus operandi* that is specific to the FOSS community, such as submitting a bug report and consulting independent user forums or mailing lists, and to evaluate the historical performance of suppliers in fixing bugs and implementing new features.

Phase 6- It is recommended that a contingency plan be prepared before deployment

A contingency plan will clarify the operational process needed to cope with problems and estimate their duration and severity in a specific operational context.

Phase 7- ***Seek approval and document lessons learned for future studies***

4.0 Discussion and Recommendations

4.1 Canadian Navy context

In the context of the HMCCS project (Halifax Modernized Command and Control Systems), the Canadian Navy is seriously considering the adoption of open standards and the use of open source software products that conform to these standards. Open standards should greatly facilitate interoperability with allies and partners, while the openness of source code offers significant benefits in developing confidence in the code and in promising more control of functionalities.
One of the first technologies to consider in the HMCCS project is the choice of an operating system. Historically, many information systems in the Canadian Navy have been built on UNIX; however, Linux can now be seen as a realistic alternative. In this chapter we will examine the support options for Linux throughout the life cycle of the HMCCS system, which is estimated to last approximately 20 years! First, we will propose a plausible way ahead for the adoption of Linux (section 4.2), look into mitigation of the associated risks (4.3), and finally provide some relative cost estimates (4.4). For readers less familiar with Linux, it may be appropriate to read the reference [13], which gives an excellent overview of this operating system and profiles the technology leaders. [14] is a Linux user survey that can provides a quantitative view of the progress of Linux in North America during 2004. Again, the focus of our discussion is the life-cycle support strategies developed and customized by the Linux communities with a view to helping the Canadian Navy adopt these strategies.

4.2 A possible way ahead for the support of Linux

(a) *Commercial Linux packages should be considered first*

This approach will give a quick start to deployment with the following significant advantages: continuity in the CF business process (a strategy similar to COTS software); stability for the organization using a mature, well-debugged OS; enhanced flexibility in sharing software with contractors and partners; and normally, some cost savings throughout this process. Support contracts can then be negotiated directly with the FOSS supplier. The Gartner report G00121102 [15] offers a helpful checklist for negotiating long-term contract protection. When tighter security is required, the guidelines in [16] and [17] offer some excellent advice. Further detailed comparisons of a few specific products are available in the Gartner reports [3, 18, 19 and 4].
At least two good vendors are available to the Canadian Navy: Red Hat Linux, Novell SUSE and many more, including major hardware manufacturers (i.e., IBM, HP, Dell, etc.) that can integrate the OS into the delivery of the basic computer. The following checklist can be used to assess various vendor support offerings:

i. How closely is the vendor tied to the development community?
ii. Is the support process internal or relayed externally to the FOSS community?
iii. Does the vendor contribute back to the community (and not just exploit it)?
iv. Can the support of this component be merged with other components (e.g., performance monitoring, user configuration management)?
v. How big is the technical team and what is the extent of their competence?

(b) *Acquiring the OS through the main C2IS contractor can improve responsiveness*

It may be preferable to obtain OS support directly from the core C2IS vendor as an integrated component, thus simplifying support management. FOSS gives this interesting flexibility in getting support for all the software components from the same software integrator including the OS. This may be preferable to ease bug fixing and improve responsiveness to cyber-attacks. Traditional approaches consisted in getting support for the OS from the hardware vendor and getting bug fixes for the application from the integrator. This appears to be less efficient for managing problems that demand a rapid response.

(c) Use other options for complementary development or as a fallback position

A classic example of a requirement that may not be available from a commercial FOSS vendor is legacy systems to cover the transition period. The use of consultants or internal specialists is often well suited to such tasks. Gartner also offers some good advice on third-party maintenance in research reports [20, 21], which include a checklist to assess third-party support.

(d) Consider joining a consortium for specialized functionality

Consortia have excellent potential to help in rapid development and add complementary functionalities economically. Interoperability requirements (with OGDs, allies, etc.) and advanced security needs are two well-known examples of requirements that need to be addressed jointly with partners.

4.3 Discussions of related risks

4.3.1 Technical risk

Changing an OS platform always entails significant risks during the migration phase, but also for life-cycle support (i.e., the focus of our discussion). Those associated with FOSS were introduced in section 3.2. Once a decision is made to change the OS, either the FOSS-based solution or the COTS-based solution will have to be scrutinized to pinpoint the key risks associated with the candidate technology. There can be no doubt both will entail risks. Clearly, even a commercial OS can carry a great deal of risk!

For example, the UNIX OS currently in use in the Canadian Navy can be replaced by a newer commercial UNIX; however, the long-term viability of these products is rather uncertain at this time, given the constantly shrinking customer base of these UNIX products. Either the supplier will abandon the support of these products or it will become prohibitively expensive to maintain them. Thus, even if the risks associated with the transition are reduced, long-term maintenance is riskier with COTS UNIX. Adopting a FOSS version of UNIX (several free UNIX alternatives now exist) will give improved complementary in the support options such as consortium-based (option 3) and self-support (option 4), which would be extremely useful when commercial support becomes less economically viable.

It is probably preferable to migrate directly to Linux and build on the most widely used (and supported) technology since (a) most of the development of new features is currently occurring on Linux-based systems (and not on UNIX!), (b) it is expected that Linux will be expanding constantly over the next ten years (see Gartner reports and other forecasting firm analyses [22, 23, 19, 24, 25, 26, 27, 28, 29, 30]), and (c) access to the source code opens up new life-cycle options (3 and 4). Of course, you must make certain that the Linux version being procured can run all the main applications of the C2IS system. Table 1 summarizes the above discussion.

Table 1
Technical Risks – Qualitative Estimates

From COTS UNIX	Migration	Short-term support	Long-term support
To COTS UNIX	*very low*	*low*	*medium to high*
To FOSS UNIX	*low*	*low to medium*	*low to medium*

To FOSS Linux	*medium*	*low*	*low*

4.3.2 Legal risks

Proprietary software and FOSS have very different legal risk profiles. An excellent discussion of the legal contexts is available in the Australian report [31]. In short, FOSS licences have imposed some constraints on the redistribution of the software in order to avoid appropriation of the software developed by (unpaid) volunteers in the commercial world. It is strongly recommended to read the AGIMO report (p. 40 to 46), which offers an excellent overview of this issue, including a decision matrix for open source licences. Many other references are available, but the AGIMO report is probably the most credible and balanced assessment available at this time.

4.4 Preliminary cost estimation

Cost estimation is another topic of heated debate. Many reports from Gartner and other forecasting firms can be found on the Internet [5, 32, 33, and 34]. One of the difficulties is to determine if such comparisons are fair (somewhat neutral – with realistic premises) or biased to support a specific perspective. Biased cost estimates can be found in both camps: the commercial world trying to discredit FOSS viability or the FOSS community promising huge savings unachievable with COTS software.

Based on the evaluation of more than 18 cost estimates from various sources, it appears clear that the difference between COTS and FOSS software is rather small (approx 10% to 15%) when short- and long-term costs are properly accounted for [35]. Linux is one of the technologies that are often studied. Sometimes COTS software is less expensive; sometimes the FOSS approach wins. Many analysts believe the TCO for both options (COTS and FOSS) will remain roughly the same for the next three years. It is also expected that the big commercial firms who enjoy great profit margins will progressively lower the cost of their support services in order to remain competitive with the emerging FOSS alternatives [22, 36, and 37]. The Gartner report G00123983 [38] offers a very enlightening discussion on FOSS TCO and related political factors that should be taken into account before switching to FOSS.

As mentioned before, the key to cost savings is to leverage the support available in the FOSS communities. The larger the community supporting FOSS, the better it is! In [39], Terry Bollinger describes practical ways for keeping the support costs down that include: (i) direct participation in the community, (ii) a mixture of public and private code factoring, (iii) allowing “time-limited” and/or “conventional” for-profit creation of new software. His discussion on “how unexpected costs arise” is very enlightening [39].

4.5 Trends and recommendations put forward for the Canadian Navy

4.5.1 Due to the actual space occupied by FOSS in modern IT infrastructures (mostly on the backbone/network/server, in software development and in security tools), and the expected expansion of FOSS that all forecasting firms predict over the next 5-10 years [22, 23, 19, 24, 25, 26, 27, 28, 29, 30], it is necessary to elaborate more precise guidelines for the support of FOSS applications in Canadian Navy IT infrastructures. This should include documentation standards. See Gartner reports [40, 41] for a good starting point on the development of best practices; see the Meta-Delta report #2858 for more general advice [42]. Gartner report G00121102 [15] offers a good checklist to help negotiate long-term contract protection and [43] for advice on cost-savings in application support.

4.5.2 With the confirmation that Linux use continues to expand in the Canadian Navy and IT organizations in general, it is recommended to educate the technical personnel that will be providing operational support for Linux and related applications, including CVS (Control

Versioning System), LSB (Linux Standard Base), POSIX, etc. See [44, 45, and 46] for a good technical introduction to Linux internals.

4.5.3 Start deployment in sub-entities of the organization rather than aiming at the whole organization. Security departments, a sub-network project or a new service being created may be a good niche for an exploratory deployment of FOSS suites, including Linux and OpenOffice.

4.5.4 A very attractive way to efficiently manage OS support is to integrate it into the management of the C2IS life cycle. Historically, OS support was provided by the hardware manufacturer, which appears to be a less efficient way of maintaining a complete information system especially when a rapid response is needed to counter a cyber-threat.

5.0 Conclusion

Software support models for FOSS and COTS are significantly different and have been a puzzlement to many users and their organizations. Often, people are afraid to take full responsibility for the support of their software, which could be an overwhelming task for many smaller organizations. However, the evolution of FOSS and their increased integration by resellers is progressively bringing the level of maturity of FOSS-based systems up to par with many commercial solutions. Furthermore, not only are the products equally good but the life-cycle support strategies can be very similar. One of the key advantages of FOSS is the richness of support options made possible by access to the source code and by the generous offerings of developer communities who constantly improve their software. Noticeably, FOSS can be maintained beyond the economic viability of commercial packages using internal resources and some contractual support, which could be a significant advantage for some military systems.

For organizations that already have an infrastructure based on UNIX, the migration to a Linux-based environment can be realistically envisioned. The migration path exists and a wide variety of support options help to diversify short- and long-term possibilities.

It appears to be a very attractive way to efficiently manage OS support and integrate it into C2IS life-cycle management. Historically, OS support was provided by the hardware manufacturers, but that appears to be a less efficient way of maintaining a complete information system.

It must be emphasized once more that the decision to integrate FOSS or COTS components must always be based on fitness for purpose and on value for money, including both short- and long-term perspectives. Our analysis confirms that much depends on the existing expertise within the organization (and close collaborators).

Increasingly, open source and commercial software will share the same ground and will “hybridize” each other in complex architectures. Of course, they are not identical and must be critically assessed in each sphere of deployment.

6.0 References

[1] Demers, David, Charpentier, Robert and Carbone, Richard (2005), Free and Open Source Software in military computing. International Command and Control Research and Technology Symposium, McLean, Virginia, June 2005

[2] Charpentier, Robert and Carbone, Richard (2004), Free and Open Source Software - Overview and Preliminary Guidelines for the Government of Canada, External Client Report ECR 2004-232, Defence R&D Canada, September 2004

[3] Weiss, G.J., Management Update: Choosing an Enterprise Linux Strategy: Novel/SUSE or Red Hat? Research ID: G00127056, Gartner, 30 March 2005

[4] Hubley, M.I., Muller, N.J., Novell SUSE Linux Operating System. Research ID: DPRO-94644, Gartner, 23 May 2005

[5] Silver, M.A., Return On Investment For Linux Desktop Migration Improves- but Not Enough for Most Users. Research ID: G00123599, Gartner, 24 June 2005

[6] Stafford, J., Linux/OSS support-part 1: Does Free Equal Shoddy? Search Enterprise Linux, 23 April 2003

[7] Stafford, J., Linux/OSS support-part 2: Ten Reasons to Turn to Open Source Support Services. Search Enterprise Linux, 30 April 2003

[8] Bartley, S., Linux/OSS support-part 3: A SYS Admin's Story. Search Enterprise Linux, 1 May 2003

[9] Stafford, J., Linux/OSS support-part 4: Step-By-Step Guide to Getting It. Search Enterprise Linux, 6 May2003

[10] Stafford, J., Linux/OSS support Weaknesses. Search Enterprise Linux, 30 April 2003

[11] Parsons, D.J., Wittman Jr, R.L., Open Source Opens Opportunities for Army Simulation System. Journal of Defense Software Engineering, January 2005

[12] Tuma, D., Open Source Software: Opportunities and Challenges. Journal of Defense Software Engineering, January 2005

[13] Hubley, M.I., Lubrano, C.R., Linux Operating System Distributions: Perspective. Research ID: DPRO-90057, Gartner, 17 August 2004

[14] Igou, B., User Survey: Linux Support Services - North America 2004. Research ID: G00127199, Gartner, 21 April 2005

[15] Disbrow, J.B., Steenstrup, K., Management Update: Negotiate Software Maintenance Terms and Conditions. Research ID: G00121102, Gartner, 26 May 2004

[16] Pescatore, J. Secure Your Linux Distribution Before Hackers Attack. Research ID: TG-23-2953, Gartner, 28 July 2004

[17] Allan, A., Certified Linux: A Seal of Approval Can't Guarantee Security. Research ID: COM-21-6272, Gartner, 23 July 2004

[18] Weiss, G.J., Choosing an Enterprise Linux Strategy: Novell/SUSE or Red Hat? Research ID G00126632, Gartner, 25 March 2005

[19] Hubley, M.I., Lubrano, C.R., Linux: What Major IT Vendors Are Doing. Research ID: DPRO-90624, Gartner, 15 November 2004

[20] Dorr, E., Third-Party Vendors Offer Alternatives for Business Application Support. Research ID: G00125704, Gartner, 30 March 2005

[21] Disbrow, J.B., Park, A.R., Be Aware of Contract Issues When Negotiating Software Escrows. Research ID: G00125669, Gartner, 7 February 2005

[22] Driver, M., Positions 2005: Open-Source Solutions Will Restructure the Software Industry. Research ID: G00126518, Gartner, 23 February 2005

[23] Weiss, G.J., Driver, M. et al., Predicts 2005: Open-Source Software Proliferates, Research ID: G00123850, Gartner, 1 November 2004

[24] Hewitt, J., Linux Making Strong Inroads in Server Market. Research ID: G00126977, Gartner, 4 April 2005

[25] Scott, D., Weiss, G.J., Linux Marches Toward Mainstream Adoption. Research ID: LE-21-5014, Gartner, 11 November 2003

[26] Weiss, G.J., Silver, M.A. et al., Hype Cycle for Linux 2005. Research ID: G00127740, Gartner, 5 July 2005

[27] Driver, M. et al., Hype Cycle for Open-Source Software 2005. Research ID: G00129254, Gartner, 20 July 2005

[28] Cearley, D.W., Fenn, J., Plummer, D.C., Gartner's Positions on the Five Hottest IT Topics and Trends in 2005. Research ID: G00125868, Gartner, 12 May 2005

[29] Weis, G.J., Chuba, M., CIO Update: Linux Survey Shows Application Interest Grows. Research ID: IGG-03172004-02, Gartner, 17 March 2004

[30] Weiss, G.J., 2004 Data Center Conference Survey Shows an Increased Linux Presence. Research ID: G00126741, Gartner, 18 March 2005

[31] Australian Government Information Management, A Guide to Open Source Software for Australian Government Agencies, Department of Finance and Administration 1 74082 084 3, April 2005

[32] Liu, V.K., Cournoyer, S. et al., Linux Proving to Be a Valuable Cost-Saving Tool in Vertical Markets. Research ID: G00121257, Gartner, 22 June 2004

[33] Silver, M.A., Examining Where Desktop Linux and Open-Source Office Products Make Sense. Research ID: G00129106, Gartner, 30 June 2005

[34] Silver, M. A., A Financial Institution Sees No ROI on Desktop Linux. Research ID: G00128022, Gartner, 24 June 2005

[35] Ferengul, C., Linux Management - Operations Excellence Infusion - Operations Strategies. Practice 2056, Meta Group, 26 June 2003

[36] Snyder, W., The Late Great Software Market - Operations Strategies. Practice 2293, Meta Group, 3 December 2004

[37] Pring, B., Brown, R.H. et al., Management Update: Predicts 2005: IT Services and Outsourcing Cut People -Costs. Research ID: G00124871, Gartner, 10 November 2004

[38] Di Maio, A., Look Beyond TCO to Judge Open-Source Software in Government, Industry. Research ID: G00123983, Gartner, 6 December 2004

[39] Bollinger T., Four Techniques for Keeping FOSS Support Costs Down, 6 March 2006 available at http://terrybollinger.com/foss/docs/Bollinger_FourTechniquesForKeepingFossSupportCostsDown.pdf

[40]Weiss, G.J., Drakos, N., A Linux/Open-Source Best-Practices Guide – The OS. Research ID: DF-19-3529, Gartner, 19 February 2003

[41] Dorr, E., Include Support and Maintenance in Competitive Evaluation of Business Applications. Research ID: G00126931, Gartner, 10 May 2005

[42] Ferengul, C., The Maturing of Linux Management. Delta 2858, Meta Group, 1 April 2004

[43] Neela, A.M., Disbrow, J.B., Three Steps to Savings in Application Support. Research ID: DF-19-9412, Gartner, 19 May 2003

[44] Hubley, M.I., Richardson, M., Linux Operating System Technology: Perspective. Research ID: DPRO-90279, Gartner, 4 June 2004

[45] Hubley, M.I., Lubrano, C.R., Linux-Based Graphical User Interfaces: Perspective. Research ID: DPRO-89832, Gartner, 19 April 2005

[46] Hubley, M.I., Lubrano, C., Debian GNU/Linux Operating System. Research ID: G00128102, Gartner, 18 June 2005

[47] Charpentier, Robert and Carbone, Richard (2005), Free and Open Source Software - FOSS in military computing. Proceedings of the Ottawa Workshop, DRDC Valcartier SL-2005-090, Defence R&D Canada, April 2005

Appendix A

(Excerpts from [47])

Talks and Presentations Given at the TTCP Workshop

(Chronological order)

1- **Dr Robert Walker**, *TTCP Joint Systems and Analysis Group Perspective*

2- **Robert Charpentier**, *Review of Workshop Objectives*

3- **Robert Charpentier**, *FOSS: Issues in Military Computing*

4- **Terry Bollinger**, *Security Implications of Using FOSS in Military Applications*

5- **Mike Sweeney**, *Using OSS in Military Information Systems: a View from the Trenches*

6- **Boyd Fletcher**, *Cross Domain Collaborative Information Environment*

7- **Frederic Michaud**, *Practical V&V Tools for C/C++ and Java – Developing Confidence in FOSS*

8- **Michel Lizotte**, *C2IS Architecture Recovery and Understanding*

9- **Casey Schaufler**, *Are Common Criteria Evaluations of FOSS Relevant?*

10- **Susan Dimitriadis**, *Creating an Institutional Repository with Dspace*

11- **Gavin Hemphill**, *OSS in DRDC Atlantic's Technology Demonstration Program*

12- **Joseph Potvin**, *The CoSE Worksite Initiative*

13- **Robert Watson**, *The Role of OSS in Effective and Rapid Technology Transfer*

14- **Michael Tiemann**, *Red Hat and FOSS Security*

15- **Iain Macleod**, *Rogue Code in FOSS: Threats and Defences*

16- **William Wolfe**, *TSWG and FOSS == Open Source Collaboration with the US Navy*

17- **Frank Meyer**, *Experiences with SELinux*

18- **Richard Taylor**, *Making the Most of FOSS*

19- **Patrick Hew**, *Irresistible FOSS meets Immovable Law-Checks – A Case Study*

Appendix B
Some Dominant FOSS Consortia

a) Linux Desktop Consortium

http://www.desktoplinuxconsortium.org/

Description: The Linux Desktop Consortium was announced by its founders on February 4, 2003 in a "pre-formation" mode. As of this writing, it does not yet have a legal structure or formal membership class. According to its initial Web site, it hopes to promote interest in and raise awareness of GNU/Linux on the desktop. Its anticipated membership will include both commercial companies as well as the various open source organizations that are developing Linux for the desktop. The new Consortium will target the needs of corporate, institutional, and home users, and intends to sponsor trade shows, conferences, and participation in Consortium-sponsored public relations activities and programs.[1]

b) Linux Standards Base (LSB)

http://www.linuxbase.org/

Description: The LSB Project develops and promotes a set of binary standards intended to increase compatibility among Linux systems (and other similar systems), and enable software applications to run on any conforming system. In addition, the LSB helps coordinate efforts to recruit software vendors to port and write products for such systems. The project as a whole operates under the auspices of the Free Standards Group.[2]

c) The Open Group

http://www.opengroup.org/

Description: The Open Group is the product of combining X/Open and the Open Software Foundation. Its current goal (as of March, 2004) is to encourage boundaryless information flow through global interoperability. The group attempts to achieve this goal through: working with customers to capture, understand and address current and emerging requirements, establish policies, and share best practices; working with suppliers, consortia and standards bodies to develop consensus and facilitate interoperability, to evolve and integrate specifications and open source technologies; offering a set of services to enhance the operational efficiency of consortia; and developing and operating an industry certification service and encouraging procurement of certified products. The group has ongoing initiatives that include active loss prevention, which is an approach aimed at protecting strategic assets by managing e-Business risk.[3]

d) Open Source Initiative (OSI)

http://www.opensource.org/

Description: The focus of the Open Source Initiative (OSI) is managing and promoting the Open Source Definition, specifically through the OSI Certified Open Source Software certification mark and program, which identifies a given

[1] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[2] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[3] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).

piece of software as complying with the OSI definition of "Open Source." It also makes copies of approved open source licenses available. Its mission includes owning and defending the Open Source Definition, developing branding programs attractive to software customers and producers, and advancing the cause of open-source software.[4]

e) Open Source And Industry Alliance (OSAIA)

http://www.osaia.org/

Description: The Open Source & Industry Alliance (OSAIA) advocates the interests of a broad array of companies, organizations, and individuals that comprise the open source community. The Alliance promotes collaborative software development and advocates on its behalf before any relevant government bodies. OSAIA promotes free and open software markets through sensible procurement, intellectual property, and competition policies, and is dedicated to the creation, use and sustainability of open source software. The Alliance believes open source licenses ensure the freedom: to employ open source software for any purpose; to study how a program works by accessing the source code so any and all can adapt it to their needs; to redistribute open source software without continuing royalty obligations to the original developer; and to improve the software and to release those improvements to the public. OSAIA is a subsidiary of the Computer and Communications Industry Association (CCIA). [5]

f) Internet Software Consortium (ISC)

http://www.isc.org

Description: The Internet Systems Consortium (ISC) focuses on developing and maintaining Open Source reference implementations of core Internet protocols. ISC was originally founded in 1994 as the Internet Software Consortium, Inc. to continue the work of maintaining and enhancing BIND (Berkeley Internet Name Domain). BIND was evolved by ISC from its original development at UC Berkeley as part of the BSD (Berkeley Software Distribution) system. At that time, IANA designated ISC as a root name server operator (first NS.ISC.ORG, then F.ROOT-SERVERS.NET) in order to support the use of BIND. Additional software systems are now hosted by ISC (INN, Lynx) and have been created by ISC (DHCP, OpenReg) to support the Internet's infrastructure. ISC has expanded operational activities beyond root name service, to include Internet hosting facilities for other open source projects (NetBSD, XFree86, kernel.org), secondary name service for more than 50 domains, and a DNS OARC (Operations, Analysis and Research Center) for monitoring and reporting of the Internet's Domain Name System.[6]

g) Open Source Software Institute (OSSI)

http://www.oss-institute.org

Description: The Open-Source Software Institute (OSSI) serves as a resource and venue for the promotion, development and implementation of open source software solutions between corporate, government and academic entities.

[4] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[5] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[6] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).

The Consortium's activities as of August 2004 include: identifying current utilization of open source software within federal, state and academic institutions; identifying opportunities for transition of closed source applications to open source software solutions; determining how current applications of closed source programs would need to be modified to operate on an open source software foundation; quantifying the costs of transition to open source software systems; developing, reviewing and/or approving training programs to enable governmental and academic staffs to transition to open source programs; receiving and implementing input from information technology sources to maximize the cost and production efficiency of the open source software programs; and/or preparing formal reports to document findings and recommendations to governmental and academic institutions.[7]

h) Open Source Consortium (OSC)

http://www.opensourceconsortium.org/oss/index.html

Description: The Open Source Consortium (OSC) provides a "proprietary-vendor free" voice for all organizations deploying or contemplating Open Source products. The OSC comprises (as of November, 2004) over 60 member companies and 400 Open Source specialists. OSC provides a range of insurance and risk solutions for the Open Source industry to help members with: negotiating profitable contracts with clients; raising capital; acquiring and retaining quality staff; investing proceeds intelligently; maximizing supply chain efficiency, and understanding and protecting intellectual property rights. OSC also works with the Open Source industry and government to educate people generally about the importance of quality and to bring about better regulation of the process of assessing quality standards in the deployment of Open Source software.[8]

i) Open Source Applications Foundation (OSAF)

http://www.osafoundation.org/

Description: The Open Source Applications Foundation (OSAF) promotes wide adoption of Open Source application software with the ultimate goal of building an innovative, sustainable alternative application. To achieve this vision, OSAF has identified long term goals (as of February, 2005) that include: designing a new application to manage email, appointments, contacts, and tasks; enabling information to be shared more easily among users; eliminating the requirement for a dedicated server or complex administration; and offering a choice of platforms and full interoperability amongst Windows, Macintosh, and Linux versions. In order to achieve these goals OSAF has identified the following immediate objectives: working within the Open Source community to extend and evolve the code base; encouraging firms to offer complementary fee-based services, support, customization and consulting; licensing fees to developers who redistribute their source code; and licensing on a fee basis to proprietary developers who do not redistribute source code.[9]

[7] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[8] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[9] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).

j) Open Source Development Lab (OSDL)

http://www.osdl.org

Description: OSDL supports enabling Linux and Linux-based applications for data center and carrier-class deployment by providing hardware for testing and development. The goal is to facilitate the addition of enterprise capabilities into Linux. Current activities (as of March, 2004) include: providing hardware resources to OSDL associates; sponsoring the Carrier Grade Linux and Data Center Linux working groups that aim to create roadmaps and specifications; participating in open source projects to develop Linux for use in telecommunication architectures and data centers; assisting in kernel testing by providing an open workbench for patch testing (the Scalable Test Platform); the creation of open source workloads; and contributing to the testing of the development kernel.[10]

k) Consumer Electronics Linux Forum (CELF)

http://www.celinuxforum.org/

Description: The Consumer Electronics Linux Forum (CELF) focuses on the advancement of Linux as an open source platform for consumer electronics (CE) devices. The CELF acts as a place to discuss various issues that are of particular importance to the CE industry. Through an open process, the CELF members attempt to clarify and codify certain requirements to be addressed in open source software. The CELF also evaluates any open source submissions with respect to their effectiveness and responsiveness to the requirements. Open source submissions accepted by the CELF Architecture Group and Steering Committee will be incorporated into the CELF source tree, which itself is open to the public. The CELF also intends to leverage the benefits of the open source community to maximize the re-use of common solutions to common problems and thereby create a foundation on which the CELF members and others can build networked products.[11]

l) Gelato Federation

http://www.gelato.org

Description: The Gelato Federation works to develop scalable, commodity software to enable researchers to advance their studies in developing technology-intensive areas, such as the life and physical sciences. Co-founded by seven research institutions, Gelato is launching an open source community initiative designed to foster the development and dissemination of focused computing solutions for researchers and associated IT staffs working on the Itanium Linux platform. Gelato's goal is to provide the research community with software downloads, including new solutions developed by Gelato member institutions and by other contributors from the greater open source community. Gelato currently (as of April, 2004) has six focus areas: compilers; parallel file systems; performance monitoring in a box; performance monitoring in a cluster; scalability in a box; and scalability in a cluster. Within these areas, Gelato is looking for ways to apply new technology and solutions to specific applications in bio-informatics, high-energy physics, and atmospheric sciences. Gelato also supplies information

[10] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[11] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).

services, such as forums and technical data, to make the Itanium Linux platform more accessible to researchers and their support staffs.[12]

m) Embedded Linux Consortium

http://www.embedded-linux.org

Description: The Embedded Linux Consortium (ELC) focuses on the advancement, promotion and standardization of Linux throughout the embedded, applied and appliance computing markets. Founded to pursue promotion of Linux for embedded applications, the Embedded Linux Consortium is also standardizing an open, unified platform specification ("ELCPS"). Members contribute to and participate in management, promotion, implementation and platform specification working group efforts.[13]

n) Free Software Foundation (FSF)

http://www.fsf.org

Description: The Free Software Foundation (FSF), established in 1985, is dedicated to promoting computer users' rights to use, study, copy, modify, and redistribute computer programs. The FSF promotes the development and use of free software, particularly the GNU operating system, used widely in its GNU/Linux variant. The FSF also helps to spread awareness of the ethical and political issues surrounding freedom in the use of software. You can read more about free software in our essays section, in the philosophy section of gnu.org, and in the pages of the independently published Free Software Magazine.[14]

o) The GNU Project

http://www.gnu.org

Description: The GNU Project was launched in 1984 to develop a complete UNIX-like operating system which is free software: the GNU system. Variants of the GNU operating system, which use the Linux kernel, are now widely used; though these systems are often referred to as “Linux,” they are more accurately called GNU/Linux systems.[15]

p) Free Standards Group (FSG)

http://www.freestandards.org/

Description: The FSG develops standards and tools to promote open source software, and coordinates testing and certification programs that verify software compliance with existing standards. FSG also promotes the use of Linux applications and platforms. Ongoing projects and workgroups of the FSG (as of March, 2004) include: the Linux Standards Base (provides a set of behavioral protocols and tests for binary compatibility among Linux distributions and enables software applications to run on any certified Linux system); openI18N (promotes language globalization of compliant distributions and applications); lanana (prevents namespace collisions by providing a responsive registration service); and openprinting (develops

[12] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[13] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[14] Description provided and quoted from the Free Software Foundation (http://www.fsf.org).
[15] Description provided and quoted from The GNU Project (http://www.gnu.org).

and promotes a set of standards that will address the needs of desktop to enterprise-ready printing).[16]

q) Open Source Technology Group

http://www.ostg.com/index.htm

Description: OSTG is the leading network of technology sites for today's IT managers and development professionals and provides a unique combination of news, original articles, downloadable resources, and community forums to help IT buyers, influencers and users make critical decisions about information technology products and services.[17]

r) IEEE Computer Society (IEEECS)

http://www.computer.org/portal/site/ieeecs/index.jsp

http://www.ieee.org

Description: The Computer Society provides technical information and services to computing professionals. It focuses on advancing the theory, practice, and application of computer and information processing technology. Through its conferences, applications-related and research-oriented journals, local and student chapters, distance learning virtual campus, technical committees, and standards working groups, the Society promotes an exchange of information, ideas, and technological innovation among its members. In addition, the Society maintains relationships with the US Computing Sciences Accreditation Board and Accreditation Board for Engineering and Technology, monitoring and evaluating curriculum accreditation guidelines.[18]

s) SourceForge

http://www.sourceforge.net

Description: SourceForge.net is the world's largest Open Source software development web site, hosting more than 100,000 projects and over 1,000,000 registered users with a centralized resource for managing projects, issues, communications, and code. SourceForge.net has the largest repository of Open Source code and applications available on the Internet, and hosts more Open Source development products than any other site or network worldwide. SourceForge.net provides a wide variety of services to projects we host, and to the Open Source community.

SourceForge.net provides free hosting to Open Source software development projects. The essence of the Open Source development model is the rapid creation of solutions within an open, collaborative environment. Collaboration within the Open Source community (developers and end-users) promotes a higher standard of quality, and helps to ensure the long-term viability of both data and applications.

SourceForge.net is owned by OSTG, Inc. ("Open Source Technology Group").[19]

---

[16] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[17] Description provided and quoted from the Open Source Technology Group (http://www.ostg.org/index.htm).
[18] Description provided and quoted from the Consortium and Standards List (http://www.consortiuminfo.org).
[19] Description provided and quoted from SourceForge (http://www.sourceforge.net).

t) Freshmeat

http://freshmeat.org

Description: Freshmeat maintains the Web's largest index of UNIX and cross-platform software, themes and related "eye-candy," and Palm OS software. Thousands of applications, which are preferably released under an open source license, are meticulously cataloged in the Freshmeat database, and links to new applications are added daily. Each entry provides a description of the software, links to download it and to obtain more information, and a history of the project's releases, so readers can keep up-to-date on the latest developments.

Freshmeat is the first stop for Linux users hunting for the software they need for work or play. It is continuously updated with the latest developments from the "release early, release often" community. In addition to providing news on new releases, Freshmeat offers a variety of original content on technical, political, and social aspects of software and programming, written by both Freshmeat readers and Free Software luminaries. The comment board attached to each page serves as a home for spirited discussion, bug reports, and technical support. An essential resource for serious developers, freshmeat.net makes it possible to keep up on who's doing what, and what everyone else thinks of it.

Freshmeat.net is owned by OSTG, Inc. ("Open Source Technology Group").[20]

u) OpenOffice

http://www.openoffice.org

Description: To create, as a community, the leading international office suite that will run on all major platforms and provide access to all functionality and data through open-component based API's and an XML-based file format.

StarDivision, the original author of the StarOffice suite of software, was founded in Germany in the mid-1980s. It was acquired by Sun Microsystems during the summer of 1999 and StarOffice 5.2 was released in June of 2000. Future versions of StarOffice software, beginning with version 6.0, have been built using the OpenOffice.org source, API's, file formats, and reference implementations. Sun continues to sponsor development on OpenOffice.org and is the primary contributor of code to OpenOffice.org. CollabNet hosts the Web site infrastructure for development of the product and helps manage the project.

The OpenOffice.org source code includes the technology which Sun Microsystems has been developing for the future versions of StarOffice(TM) software. The source is written in C++ and delivers language-neutral and scriptable functionality, including Java(TM) API's. This source technology introduces the next-stage architecture, allowing use of the suite as separate applications or as embedded components in other

[20] Description provided and quoted from Freshmeat (http://freshmeat.net).

applications. Numerous other features are also present including XML-based file formats and other resources.[21]

N.B.: All consortia descriptions are made available from the aforementioned Web sites. Descriptions may have been modified to correct errors in syntax, logic, grammar, or spelling for purposes of readability and clarity. Every effort has been made to ensure accuracy and similarity to the original content copied from the aforementioned Web sites. Descriptions were quoted rather than rewritten/paraphrased to preserve the original sense and content of the description.

[21] Description provided and quoted from OpenOffice (http://www.openoffice.org).